Formalizing Bachmair and Ganzinger's Ordered Resolution Prover

Anders Schlichtkrull, Jasmin Christian Blanchette, Dmitriy Traytel, Uwe Waldmann

Formalizing Bachmair and Ganzinger's Ordered Resolution Prover

Anders Schlichtkrull[1(✉)], Jasmin Christian Blanchette[2,3], Dmitriy Traytel[4], and Uwe Waldmann[3]

[1] DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
andschl@dtu.dk
[2] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
[3] Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
[4] Institute of Information Security, Department of Computer Science, ETH Zürich, Zurich, Switzerland

Abstract. We present a formalization of the first half of Bachmair and Ganzinger's chapter on resolution theorem proving in Isabelle/HOL, culminating with a refutationally complete first-order prover based on ordered resolution with literal selection. We develop general infrastructure and methodology that can form the basis of completeness proofs for related calculi, including superposition. Our work clarifies several of the fine points in the chapter's text, emphasizing the value of formal proofs in the field of automated reasoning.

1 Introduction

Much research in automated reasoning amounts to metatheoretical arguments, typically about the soundness and completeness of logical inference systems or the termination of theorem proving processes. Often the proofs contain more insights than the systems or processes themselves. For example, the superposition calculus rules [2], with their many side conditions, look rather arbitrary, whereas in the completeness proof the side conditions emerge naturally from the model construction. And yet, despite being crucial to our field, today such proofs are usually carried out without tool support beyond TEX.

We believe proof assistants are becoming mature enough to help. In this paper, we present a formalization, developed using the Isabelle/HOL system [16], of a first-order prover based on ordered resolution with literal selection. We follow Bachmair and Ganzinger's account [3] from Chapter 2 of the *Handbook of Automated Reasoning*, which we will simply refer to as "the chapter." Our formal development covers the refutational completeness of two resolution calculi for ground (i.e., variable-free) clauses and general infrastructure for theorem proving processes and redundancy, culminating with a completeness proof for a first-order prover expressed as transition rules operating on triples of clause sets. This material corresponds to the chapter's first four sections.

From the perspective of automated reasoning, increased trustworthiness of the results is an obvious benefit of formal proofs. But formalizing also helps clarify arguments, by exposing and explaining difficult steps. Making theorem statements (including definitions and hypotheses) precise can be a huge gain for communicating results. Moreover, a formal proof can tell us exactly where hypotheses and lemmas are used. Once we have created a library of basic results and a methodology, we will be in a good

position to study extensions and variants. Given that automatic theorem provers are integrated in modern proof assistants, there is also an undeniable thrill in applying these tools to reason about their own metatheory. From the perspective of interactive theorem proving, formalization work constitutes a case study in the use of a proof assistant. It gives us, as developers and users of such a system, an opportunity to experiment, contribute to lemma libraries, and get inspiration for new features and improvements.

Our motivation for choosing Bachmair and Ganzinger's chapter is manyfold. The text is a standard introduction to superposition-like calculi (together with *Handbook* Chapters 7 [14] and 27 [26]). It offers perhaps the most detailed treatment of the lifting of a resolution-style calculus's static completeness to a saturation prover's dynamic completeness. It introduces a considerable amount of general infrastructure, including different types of inference systems (sound, reductive, counterexample-reducing, etc.), theorem proving processes, and an abstract notion of redundancy. The resolution calculus, extended with a term order and literal selection, captures most of the insights underlying ordered paramodulation and superposition, but with a simple notion of model.

The chapter's level of rigor is uneven, as shown by the errors and imprecisions revealed by our formalization. We will see that the main completeness result does not hold, due to the improper treatment of self-inferences. Naturally, our objective is not to diminish Bachmair and Ganzinger's outstanding achievements, which include the development of superposition; rather, it is to demonstrate that even the work of some of the most celebrated researchers in our field can benefit from formalization. Our view is that formal proofs can be used to complement and improve their informal counterparts.

This work is part of the IsaFoL (Isabelle Formalization of Logic) project,[1] which aims at developing a library of results about logical calculi. The Isabelle files are available in the *Archive of Formal Proofs* (AFP).[2] They amount to about 8000 lines of source text. Below we provide implicit hyperlinks from theory and lemma names. A better way to study the theory files, however, is to open them in Isabelle/jEdit [28]. We used Isabelle version 2017, but the AFP is continuously updated to track Isabelle's evolution. Due to lack of space, we assume the reader has some familiarity with the chapter's content. An extended version of this paper is available as a technical report [21].

2 Preliminaries

Ordered resolution depends on little background metatheory. Much of it, concerning partial and total orders, well-foundedness, and finite multisets, is provided by standard Isabelle libraries. We also need literals, clauses, models, terms, and substitutions.

Clauses and Models. We use the same library of clauses (`Clausal_Logic.thy`) as for the verified SAT solver by Blanchette et al. [6], which is also part of IsaFoL. Atoms are represented by a type variable $'a$, which can be instantiated by arbitrary concrete types—e.g., numbers or first-order terms. A literal, of type $'a$ *literal* (where the type constructor is written in ML-style postfix syntax), can be of the form $\mathsf{Pos}\ A$ or $\mathsf{Neg}\ A$, where $A :: 'a$ is an atom. The literal order $>$ extends a fixed atom order $>$ by comparing polarities to break ties, with $\mathsf{Neg}\ A > \mathsf{Pos}\ A$. A clause is a finite multiset of literals,

[1] `https://bitbucket.org/isafol/isafol/wiki/Home`

[2] `https://devel.isa-afp.org/entries/Ordered_Resolution_Prover.html`

'a clause = *'a literal multiset*, where *multiset* is the Isabelle type constructor of finite multisets. Thus, the clause $A \vee B$, where A and B are atoms, is identified with the multiset $\{A, B\}$; the clause $C \vee D$, where C and D are clauses, is $C \uplus D$; and the empty clause $\bot$ is $\{\}$. The clause order is the multiset extension of the literal order.

A Herbrand interpretation I is a value of type *'a set*, specifying which ground atoms are true (`Herbrand_Interpretation.thy`). The "models" operator $\models$ is defined on atoms, literals, clauses, sets, and multisets of clauses; for example, $I \models C \Leftrightarrow \exists L \in C.\ I \models L$. Satisfiability of a set or multiset of clauses N is defined by $\mathsf{sat}\ N \Leftrightarrow \exists I.\ I \models N$.

Multisets are central to our development. Isabelle provides a multiset library, but it is much less developed than those of sets and lists. As part of IsaFoL, we have already extended it considerably and implemented further additions in a separate file (`Multiset_More.thy`). Some of these, notably a plugin for Isabelle's simplifier to apply cancellation laws, are described in a recent paper [7, Section 3].

Terms and Substitutions. The IsaFoR (Isabelle Formalization of Rewriting) library—an inspiration for IsaFoL—contains a definition of first-order terms and results about substitutions and unification [23]. It makes sense to reuse this functionality. A practical issue is that most of IsaFoR is not accessible from the AFP.

Resolution depends only on basic properties of terms and atoms, such as the existence of most general unifiers (MGUs). We exploit this to keep the development parameterized by a type of atoms $'a$ and an abstract type of substitutions $'s$, through Isabelle locales [4] (`Abstract_Substitution.thy`). A locale represents a module parameterized by types and terms that satisfy some assumptions. Inside the locale, we can refer to the parameters and assumptions in definitions, lemmas, and proofs. The basic operations provided by our locale are application ($\cdot :: 'a \Rightarrow 's \Rightarrow 'a$), identity ($\mathsf{id} :: 's$), and composition ($\circ :: 's \Rightarrow 's \Rightarrow 's$), about which some assumptions are made (e.g., $A \cdot \mathsf{id} = A$). Substitution is lifted to literals, clauses, sets of clauses, and so on. Many other operations can be defined in terms of the primitives—for example, $\mathsf{is\_ground}\ A \Leftrightarrow \forall \sigma.\ A = A \cdot \sigma$.

To complete our development and ensure that our assumptions are legitimate, we instantiate the locale's parameters with IsaFoR types and operations and discharge its assumptions (`IsaFoR_Term.thy`). This bridge is currently hosted outside the AFP.

3 Refutational Inference Systems

In their Section 2.4, Bachmair and Ganzinger introduce basic conventions for refutational inference systems. In Section 3, they present two ground resolution calculi and prove them refutationally complete in Theorems 3.9 and 3.16. In Section 4.2, they introduce a notion of counterexample-reducing inference system and state Theorem 4.4 as a generalization of Theorems 3.9 and 3.16 to all such systems. For formalization, two courses of actions suggest themselves: follow the book closely and prove the three theorems separately, or focus on the most general result. We choose the latter, as being more consistent with the goal of providing a well-designed, reusable library.

We collect the abstract hierarchy of inference systems in a single Isabelle theory file (`Inference_System.thy`). An inference, of type *'a inference*, is a triple $(\mathcal{C}, D, E)$ that consists of a multiset of side premises $\mathcal{C}$, a main premise D, and a conclusion E. An inference system, or calculus, is a possibly infinite set of inferences:

locale *inference_system* = **fixes** $\Gamma :: 'a$ *inference set*

We use an Isabelle locale to fix, within a named context (*inference_system*), a set Γ of inferences between clauses over atom type $'a$. Inside the locale, we define a function infers_from that, given a clause set N, returns the subset of Γ inferences whose premises all belong to N. A satisfiability-preserving (or consistency-preserving) inference system enriches the inference system locale with an assumption, whereas sound systems are characterized by a different assumption:

locale *sat_preserving_inference_system* = *inference_system* +
  assumes sat $N \Longrightarrow$ sat $(N \cup$ concl_of ` infers_from $N)$

locale *sound_inference_system* = *inference_system* +
  assumes $(\mathcal{C}, D, E) \in \Gamma \Longrightarrow I \models \mathcal{C} \cup \{D\} \Longrightarrow I \models E$

The notation f ` X above stands for the image of the set or multiset X under function f.

Soundness is a stronger requirement than satisfiability preservation. In Isabelle:

sublocale *sound_inference_system* < *sat_preserving_inference_system*

This command emits a proof goal stating that *sound_inference_system*'s assumption implies *sat_preserving_inference_system*'s. Afterwards, all the definitions and lemmas about satisfiability-preserving calculi become available about sound ones.

In reductive inference systems (*reductive_inference_system*), the conclusion of each inference is smaller than the main premise according to the clause order. A related notion, the counterexample-reducing inference systems, is specified as follows:

locale *counterex_reducing_inference_system* = *inference_system* +
  fixes I_of :: $'a$ *clause set* $\Rightarrow$ $'a$ *set*
  assumes $\{\} \notin N \Longrightarrow D \in N \Longrightarrow$ I_of $N \not\models D \Longrightarrow$
    $(\forall C \in N.$ I_of $N \not\models C \Longrightarrow D \leq C) \Longrightarrow$
    $\exists \mathcal{C} \subseteq N. \exists E.$ I_of $N \models \mathcal{C} \wedge (\mathcal{C}, D, E) \in \Gamma \wedge$ I_of $N \not\models E \wedge E < D$

The "model functor" parameter I_of maps clause sets to candidate models. The assumption is that for any set N that does not contain $\{\}$ (i.e., $\bot$), if $D \in N$ is the smallest counterexample—the smallest clause in N falsified by I_of N—we can derive a smaller counterexample E using an inference from clauses in N. This property is useful because if N is saturated (i.e., closed under Γ), we must have $E \in N$, violating D's minimality:

theorem *saturated_model*: saturated $N \Longrightarrow \{\} \notin N \Longrightarrow$ I_of $N \models N$

corollary *saturated_complete*: saturated $N \Longrightarrow \neg$ sat $N \Longrightarrow \{\} \in N$

Bachmair and Ganzinger claim that compactness of clausal logic follows from the refutational completeness of ground resolution (Theorem 3.12), although they give no justification. Our argument relies on an inductive definition of saturation of a set of clauses: saturate :: $'a$ *clause set* $\Rightarrow$ $'a$ *clause set*. Most of the work goes into proving this key lemma, by rule induction on the saturate function:

lemma *saturate_finite*: $C \in$ saturate $N \Longrightarrow \exists M \subseteq N.$ finite $M \wedge C \in$ saturate M

The interesting case is when $C = \bot$. We establish compactness in a locale that combines *counterex_reducing_inference_system* and *sound_inference_system*:

theorem *clausal_logic_compact*: $\neg$ sat $N \Leftrightarrow \exists M \subseteq N.$ finite $M \wedge \neg$ sat M

4 Ground Resolution

A useful strategy for establishing properties of first-order calculi is to initially restrict our attention to ground calculi and then to lift the results to first-order formulas containing terms with variables. Accordingly, the chapter's Section 3 presents two ground calculi: a simple binary resolution calculus and an ordered resolution calculus with literal selection. Both consist of a single resolution rule, with built-in positive factorization. Most of the explanations and proofs concern the simpler calculus. To avoid duplication, we factor out the candidate model construction (`Ground_Resolution_Model.thy`). We then define the two calculi and prove that they are sound and reduce counterexamples (`Unordered_Ground_Resolution.thy`, `Ordered_Ground_Resolution.thy`).

Candidate Models. Refutational completeness is proved by exhibiting a model for any saturated clause set N that does not contain $\bot$. The model is constructed incrementally, one clause $C \in N$ at a time, starting with an empty Herbrand interpretation. The idea appears to have originated with Brand [10] and Zhang and Kapur [29].

Bachmair and Ganzinger introduce two operators to build the candidate model: I_C denotes the current interpretation before considering C, and ε_C denotes the set of (zero or one) atoms added, or *produced*, to ensure that C is satisfied. The candidate model construction is parameterized by a literal selection function $S :: 'a\ clause \Rightarrow 'a\ clause$. We also fix a clause set N. Then we define two operators corresponding to ε_C and I_C:

function production :: *'a clause* $\Rightarrow$ *'a set* **where**
production $C = \{A \mid C \in N \wedge C \neq \{\} \wedge \mathsf{Max}\ C = \mathsf{Pos}\ A \wedge \big(\bigcup_{D<C} \mathsf{production}\ D\big) \not\models C \wedge S\ C = \{\}\}$

definition interp :: *'a clause* $\Rightarrow$ *'a set* **where**
interp $C = \bigcup_{D<C} \mathsf{production}\ D$

To ensure monotonicity of the construction, any produced atom must be maximal in its clause. Moreover, productive clauses may not contain selected literals. In the chapter, ε_C and I_C are expressed in terms of each other. We simplified the definition by inlining I_C in ε_C, so that only ε_C is recursive. Since the recursive calls operate on clauses D that are smaller with respect to a well-founded order, the definition is accepted. Bachmair and Ganzinger's I^C and I_N operators are introduced as abbreviations: $\mathsf{Interp}\ C = \mathsf{interp}\ C \cup \mathsf{production}\ C$ and $\mathsf{INTERP} = \bigcup_{C \in N} \mathsf{production}\ C$.

We then prove a host of lemmas about these concepts. Lemma 3.4 amounts to six monotonicity properties, including these:

lemma *interp_imp_Interp*: $C \leq D \Longrightarrow D \leq D' \Longrightarrow \mathsf{interp}\ D \models C \Longrightarrow \mathsf{Interp}\ D' \models C$
lemma *Interp_imp_INTERP*: $C \leq D \Longrightarrow \mathsf{Interp}\ D \models C \Longrightarrow \mathsf{INTERP} \models C$

Lemma 3.3, whose proof depends on monotonicity, is better proved *after* 3.4:

lemma *productive_imp_INTERP*: $\mathsf{production}\ C \neq \{\} \Longrightarrow \mathsf{INTERP} \models C$

A more serious oddity is Lemma 3.7. Using our notations, it can be stated as $D \in N \Longrightarrow C \neq D \Longrightarrow (\forall D' < D.\ \mathsf{Interp}\ D' \models C) \Longrightarrow \mathsf{interp}\ D \models D'$. However, the last occurrence of D' is clearly wrong—the context suggests C instead. Even after this amendment, we have a counterexample, corresponding to a gap in the proof: $D = \{\}$, $C = \{\mathsf{Pos}\ A\}$, and $N = \{D, C\}$. Since this "lemma" is not actually used, we can simply ignore it.

Unordered Resolution. The unordered ground resolution calculus consists of a single binary inference rule, with the side premise $C \vee A \vee \cdots \vee A$, the main premise $\neg A \vee D$, and the conclusion $C \vee D$. Formally, this rule is captured by a predicate:

inductive unord_resolve :: *'a clause* ⇒ *'a clause* ⇒ *'a clause* ⇒ *bool* **where**
unord_resolve $(C \uplus \mathsf{replicate}\ (n+1)\ (\mathsf{Pos}\ A))\ (\{\mathsf{Neg}\ A\} \uplus D)\ (C \uplus D)$

To prove completeness, it suffices to show that the calculus reduces counterexamples (Theorem 3.8). By instantiating the *sound_inference_system* and *counterex_reducing_inference_system* locales, we obtain refutational completeness (Theorem 3.9 and Corollary 3.10) and compactness of clausal logic (Theorem 3.12).

Ordered Resolution with Selection. Ordered ground resolution consists of a single rule, ord_resolve. Like unord_resolve, it is sound and counterexample-reducing (Theorem 3.15). Moreover, it is reductive (Lemma 3.13): the conclusion is always smaller than the main premise according to the clause order. The rule is given as

$$\frac{C_1 \vee A_1 \vee \cdots \vee A_1 \quad \cdots \quad C_n \vee A_n \vee \cdots \vee A_n \quad \neg A_1 \vee \cdots \vee \neg A_n \vee D}{C_1 \vee \cdots \vee C_n \vee D}$$

with multiple side conditions whose role is to prune the search space and to make the rule reductive. In Isabelle, we represent the n side premises by three parallel lists of length n: $C\mathcal{A}s$ gives the entire clauses, whereas Cs and $\mathcal{A}s$ store the C_i and the $\mathcal{A}_i = A_i \vee \cdots \vee A_i$ parts separately. In addition, As is the list $[A_1, \ldots, A_n]$. The following inductive definition captures the rule formally:

inductive ord_resolve :: *'a clause list* ⇒ *'a clause* ⇒ *'a clause* ⇒ *bool* **where**
$|C\mathcal{A}s| = n \Longrightarrow |Cs| = n \Longrightarrow |\mathcal{A}s| = n \Longrightarrow |As| = n \Longrightarrow n \neq 0 \Longrightarrow$
$(\forall i < n.\ C\mathcal{A}s\,!\,i = Cs\,!\,i \uplus \mathsf{Pos}\ `\ \mathcal{A}s\,!\,i) \Longrightarrow (\forall i < n.\ \mathcal{A}s\,!\,i \neq \{\}) \Longrightarrow$
$(\forall i < n.\ \forall A \in \mathcal{A}s\,!\,i.\ A = As\,!\,i) \Longrightarrow \mathsf{eligible}\ As\ (D \uplus \mathsf{Neg}\ `\ \mathsf{mset}\ As) \Longrightarrow$
$(\forall i < n.\ \mathsf{strict\_max\_in}\ (As\,!\,i)\ (Cs\,!\,i)) \Longrightarrow (\forall i < n.\ S\ (C\mathcal{A}s\,!\,i) = \{\}) \Longrightarrow$
ord_resolve $C\mathcal{A}s\ (D \uplus \mathsf{Neg}\ `\ \mathsf{mset}\ As)\ ((\bigcup \mathsf{mset}\ Cs) \uplus D)$

The $xs\,!\,i$ operator returns the $(i+1)$st element of xs, and mset converts a list to a multiset. Initially, we tried storing the n premises in a multiset, since their order is irrelevant. However, due to the permutative nature of multisets, there can be no such things as "parallel multisets"; the alternative, a single multiset of tuples, is very unwieldy.

Formalization revealed an error and a few ambiguities in the rule's statement. References to $S(D)$ in the side conditions should have been to $S(\neg A_1 \vee \cdots \vee \neg A_n \vee D)$. The ambiguities are discussed in our technical report [21, Appendix A].

5 Theorem Proving Processes

In their Section 4, Bachmair and Ganzinger switch to a dynamic view of saturation: from clause sets closed under inferences to theorem proving processes that start with a clause set N_0 and keep deriving new clauses until no inferences are possible. Redundant clauses can be deleted at any point, and redundant inferences need not be performed.

A derivation performed by a proving process is a possibly infinite sequence $N_0 \rhd N_1 \rhd N_2 \rhd \cdots$, where $\rhd$ relates clause sets (`Proving_Process.thy`). In Isabelle, such sequences are captured by lazy lists, a codatatype [5] generated by LNil :: *'a llist* and

LCons :: $'a \Rightarrow 'a\ llist \Rightarrow 'a\ llist$, and equipped with lhd ("head") and ltl ("tail") selectors that extract LCons's arguments. The coinductive predicate chain checks that its argument is a nonempty lazy list whose elements are linked by a binary predicate R:

coinductive chain :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow 'a\ llist \Rightarrow bool$ **where**
  chain R (LCons x LNil)
| chain $R\ xs \Longrightarrow R\ x$ (lhd xs) $\Longrightarrow$ chain R (LCons $x\ xs$)

A derivation is a lazy list Ns of clause sets satisfying the chain predicate with $R = \rhd$. Derivations depend on a redundancy criterion presented as two functions, $\mathcal{R}_\mathcal{F}$ and $\mathcal{R}_\mathcal{I}$:

locale *redundancy_criterion* = *inference_system* +
  fixes $\mathcal{R}_\mathcal{F} :: 'a\ clause\ set \Rightarrow 'a\ clause\ set$ **and** $\mathcal{R}_\mathcal{I} :: 'a\ clause\ set \Rightarrow 'a\ inference\ set$
  assumes $\mathcal{R}_\mathcal{I}\ N \subseteq \Gamma$ **and** sat $(N \setminus \mathcal{R}_\mathcal{F}\ N) \Longrightarrow$ sat N **and**
    $N \subseteq N' \Longrightarrow \mathcal{R}_\mathcal{F}\ N \subseteq \mathcal{R}_\mathcal{F}\ N' \wedge \mathcal{R}_\mathcal{I}\ N \subseteq \mathcal{R}_\mathcal{I}\ N'$ **and**
    $N' \subseteq \mathcal{R}_\mathcal{F}\ N \Longrightarrow \mathcal{R}_\mathcal{F}\ N \subseteq \mathcal{R}_\mathcal{F}\ (N \setminus N') \wedge \mathcal{R}_\mathcal{I}\ N \subseteq \mathcal{R}_\mathcal{I}\ (N \setminus N')$

By definition, a transition from M to N is possible if the only new clauses added are conclusions of inferences from M and any deleted clauses would be redundant in N:

$$M \rhd N \Leftrightarrow N \setminus M \subseteq \text{concl\_of `` infers\_from } M \wedge M \setminus N \subseteq \mathcal{R}_\mathcal{F}\ N$$

This rule combines deduction (the addition of inferred clauses) and deletion (the removal of redundant clauses) in a single transition. The chapter keeps the two operations separated, but this is problematic, as we will see in Section 7.

A key concept to connect static and dynamic completeness is that of the set of persistent clauses, or limit: $N_\infty = \bigcup_i \bigcap_{j \geq i} N_j$. These are the clauses that belong to all clause sets except for at most a finite prefix of the sequence N_i. We also need the supremum of a sequence, $\bigcup_i N_i$. We introduce these missing functions (`Lazy_List_Liminf.thy`):

$$\text{Liminf}\ xs = \bigcup_{i<|xs|} \bigcap_{j:i\leq j<|xs|} xs\,!\,j \qquad\qquad \text{Sup}\ xs = \bigcup_{i<|xs|} xs\,!\,i$$

When interpreting the notation $\bigcup_i \bigcap_{j\geq i} N_j$ for the case of a finite sequence of length n, it is crucial to use the right upper bounds, namely $i, j < n$. For i, the danger is subtle: if $i \geq n$, then $\bigcap_{j:i\leq j<n} N_j$ collapses to the trivial infimum $\bigcap_{j\in\{\}} N_j$, i.e., the set of all clauses.

Lemma 4.2 connects the redundant clauses and inferences at the limit to those of the supremum, and the satisfiability of the limit to that of the initial clause set. Formally:

lemma *Rf_limit_Sup*: chain $(\rhd)\ Ns \Longrightarrow \mathcal{R}_\mathcal{F}$ (Liminf Ns) $= \mathcal{R}_\mathcal{F}$ (Sup Ns)
lemma *Ri_limit_Sup*: chain $(\rhd)\ Ns \Longrightarrow \mathcal{R}_\mathcal{I}$ (Liminf Ns) $= \mathcal{R}_\mathcal{I}$ (Sup Ns)
lemma *sat_limit_iff*: chain $(\rhd)\ Ns \Longrightarrow \big($sat (Liminf Ns) $\Leftrightarrow$ sat (lhd Ns)$\big)$

In the chapter, the proof relies on "the soundness of the inference system," contradicting the claim that "we will only consider consistency-preserving inference systems" [2, Section 2.4]. Thanks to Isabelle, we now know that soundness is unnecessary.

Next, we show that the limit is saturated, under some assumptions and for a relaxed notion of saturation. A clause set N is saturated up to redundancy if all inferences from nonredundant clauses in N are redundant:

$$\text{saturated\_upto}\ N \Leftrightarrow \text{infers\_from}\ (N \setminus \mathcal{R}_\mathcal{F}\ N) \subseteq \mathcal{R}_\mathcal{I}\ N$$

The limit is saturated for fair derivations, defined by fair_clss_seq $Ns \Leftrightarrow$ concl_of ` infers_from $N' \setminus \mathcal{R}_\mathcal{I}\ N' \subseteq$ Sup $Ns \cup \mathcal{R}_\mathcal{F}$ (Sup Ns) with $N' =$ Liminf $Ns \setminus \mathcal{R}_\mathcal{F}$ (Liminf Ns).

The criterion must also be effective, meaning $\gamma \in \Gamma \Longrightarrow \mathsf{concl\_of}\ \gamma \in N \cup \mathcal{R}_{\mathcal{F}}\, N \Longrightarrow \gamma \in \mathcal{R}_{\mathcal{I}}\, N$. Under these assumptions, we have Theorem 4.3:

theorem *fair_derive_saturated_upto*:
$\mathsf{chain}\ (\rhd)\ Ns \Longrightarrow \mathsf{fair\_clss\_seq}\ Ns \Longrightarrow \mathsf{saturated\_upto}\ (\mathsf{Liminf}\ Ns)$

The standard redundancy criterion is an instance of the framework. It relies on a counterexample-reducing inference system Γ (`Standard_Redundancy.thy`):

$$\mathcal{R}_{\mathcal{F}}\, N = \{C \mid \exists \mathcal{D} \subseteq N.\ (\forall I.\ I \vDash \mathcal{D} \Longrightarrow I \vDash C) \wedge \forall D' \in \mathcal{D}.\ D' < C\}$$
$$\mathcal{R}_{\mathcal{I}}\, N = \{(\mathcal{C}, D, E) \in \Gamma \mid \exists \mathcal{D} \subseteq N.\ (\forall I.\ I \vDash \mathcal{D} \uplus \mathcal{C} \Longrightarrow I \vDash E) \wedge \forall D' \in \mathcal{D}.\ D' < D\}$$

Standard redundancy qualifies as *effective_redundancy_criterion*. In the chapter, this is stated as Theorems 4.7 and 4.8, which depend on two auxiliary properties, Lemmas 4.5 and 4.6. The main result, Theorem 4.9, is that counterexample-reducing calculi are refutationally complete also under the application of standard redundancy:

theorem *saturated_upto_complete*: $\mathsf{saturated\_upto}\ N \Longrightarrow (\neg\ \mathsf{sat}\ N \Leftrightarrow \{\} \in N)$

The informal proof of Lemma 4.6 applies Lemma 4.5 in a seemingly impossible way, confusing redundant clauses and redundant inferences and exploiting properties that appear only in the first lemma's proof. Our solution is to generalize the core argument into a lemma and apply it to prove Lemmas 4.5 and 4.6. Incidentally, the informal proof of Theorem 4.9 also needlessly invokes Lemma 4.5.

Finally, given a redundancy criterion $(\mathcal{R}_{\mathcal{F}}, \mathcal{R}_{\mathcal{I}})$ for Γ, its standard extension for $\Gamma' \supseteq \Gamma$ is defined as $(\mathcal{R}_{\mathcal{F}}, \mathcal{R}'_{\mathcal{I}})$, where $\mathcal{R}'_{\mathcal{I}}\, N = \mathcal{R}_{\mathcal{I}}\, N \cup (\Gamma' \setminus \Gamma)$ (`Proving_Process.thy`). The standard extension preserves effectiveness, saturation up to redundancy, and fairness.

6 First-Order Resolution

The chapter's Section 4.3 presents a first-order version of the ordered resolution rule and a first-order prover, RP, based on that rule. The first step towards lifting the completeness of ground resolution is to show that we can lift individual ground resolution inferences (`FO_Ordered_Resolution.thy`).

Inference Rule. First-order ordered resolution consists of the single rule

$$\frac{C_1 \vee A_{11} \vee \cdots \vee A_{1k_1} \quad \cdots \quad C_n \vee A_{n1} \vee \cdots \vee A_{nk_n} \quad \neg A_1 \vee \cdots \vee \neg A_n \vee D}{C_1 \cdot \sigma \vee \cdots \vee C_n \cdot \sigma \vee D \cdot \sigma}$$

where σ is the (canonical) MGU that solves all unification problems $A_{i1} \stackrel{?}{=} \cdots \stackrel{?}{=} A_{ik_i} \stackrel{?}{=} A_i$, for $1 \leq i \leq n$. As expected, the rule has several side conditions. The Isabelle representation of this rule is based on that of its ground counterpart, generalized to apply σ:

inductive ord_resolve :: *'a clause list* $\Rightarrow$ *'a clause* $\Rightarrow$ *'s* $\Rightarrow$ *'a clause* $\Rightarrow$ *bool* **where**
$|C\mathcal{A}s| = n \Longrightarrow |Cs| = n \Longrightarrow |\mathcal{A}s| = n \Longrightarrow |As| = n \Longrightarrow n \neq 0 \Longrightarrow$
$(\forall i < n.\ C\mathcal{A}s\,!\,i = Cs\,!\,i \uplus \mathsf{Pos}\ `\ \mathcal{A}s\,!\,i) \Longrightarrow (\forall i < n.\ \mathcal{A}s\,!\,i \neq \{\}) \Longrightarrow$
$\mathsf{Some}\ \sigma = \mathsf{mgu}\ (\mathsf{set\_mset}\ `\ \mathsf{set}\ (\mathsf{map2}\ \mathsf{add\_mset}\ As\ \mathcal{A}s)) \Longrightarrow$
$\mathsf{eligible}\ \sigma\ As\ (D \uplus \mathsf{Neg}\ `\ \mathsf{mset}\ As) \Longrightarrow$
$(\forall i < n.\ \mathsf{strict\_max\_in}\ (As\,!\,i \cdot \sigma)\ (Cs\,!\,i \cdot \sigma)) \Longrightarrow (\forall i < n.\ S\ (C\mathcal{A}s\,!\,i) = \{\}) \Longrightarrow$
$\mathsf{ord\_resolve}\ C\mathcal{A}s\ (D \uplus \mathsf{Neg}\ `\ \mathsf{mset}\ As)\ \sigma\ (((\bigcup \mathsf{mset}\ Cs) \uplus D) \cdot \sigma)$

The rule as stated is incomplete; for example, $\mathsf{p}(x)$ and $\neg\,\mathsf{p}(\mathsf{f}(x))$ cannot be resolved because x and $\mathsf{f}(x)$ are not unifiable. In the chapter, the authors circumvent this issue by stating, "We also implicitly assume that different premises and the conclusion have no variables in common; variables are renamed if necessary." For the formalization, we first considered enforcing the invariant that all derived clauses use mutually disjoint variables, but this does not help when a clause is repeated in an inference's premises. Instead, we rely on a predicate ord_resolve_rename, based on ord_resolve, that standardizes the premises apart. The renaming is performed by a function called renamings_apart :: *'a clause list* ⇒ *'s list* that, given a list of clauses, returns a list of corresponding substitutions to apply. This function is part of the abstract interface for terms and substitutions (which we presented in Section 2) and is implemented using IsaFoR.

Lifting Lemma. To lift ground inferences to the first-order level, we consider a set of clauses M and introduce an adjusted version S_M of the selection function S. This new selection function depends on both S and M and works in such a way that any ground instance inherits the selection of at least one of the nonground clauses of which it is an instance. This property is captured formally as

lemma *S_M_grounding_of_clss*:
$C \in \mathsf{grounding\_of}\ M \Longrightarrow \exists D \in M.\, \exists \sigma.\ C = D \cdot \sigma \wedge S_M\, C = S\, D \cdot \sigma$

where grounding_of M is the set of ground instances of a set of clauses M.

The lifting lemma, Lemma 4.12, states that whenever there exists a ground inference of E from clauses belonging to grounding_of M, there exists a (possibly) more general inference from clauses belonging to M:

lemma *ord_resolve_rename_lifting*:
$(\forall \rho\ C.\ \mathsf{is\_renaming}\ \rho \Longrightarrow S\ (C \cdot \rho) = S\ C \cdot \rho) \Longrightarrow$
$\mathsf{ord\_resolve}\ S_M\ C\mathcal{A}s\ DA\ \mathcal{A}s\ As\ \sigma\ E \Longrightarrow$
$\{DA\} \cup \mathsf{set}\ CAs \subseteq \mathsf{grounding\_of}\ M \Longrightarrow$
$\exists \eta s\ \eta\ \theta\ C\mathcal{A}s_0\ DA_0\ \mathcal{A}s_0\ As_0\ E_0\ \tau.$
$\quad \mathsf{ord\_resolve\_rename}\ S\ C\mathcal{A}s_0\ DA_0\ \mathcal{A}s_0\ As_0\ \tau\ E_0 \wedge$
$\quad C\mathcal{A}s_0 \cdot \eta s = C\mathcal{A}s \wedge DA_0 \cdot \eta = DA \wedge E_0 \cdot \theta = E \wedge \{DA_0\} \cup \mathsf{set}\ C\mathcal{A}s_0 \subseteq M$

The informal proof of this lemma consists of two sentences spanning four lines of text. In Isabelle, these two sentences translate to 75 lines and 400 lines, respectively, excluding auxiliary lemmas. Our proof involves six steps:

1. Obtain a list of first-order clauses $C\mathcal{A}s_0$ and a first-order clause DA_0 that belong to M and that generalize $C\mathcal{A}s$ and DA with substitutions ηs and η, respectively.
2. Choose atoms $\mathcal{A}s_0$ and As_0 in the first-order clauses on which to resolve.
3. Standardize $C\mathcal{A}s_0$ and DA_0 apart, yielding $C\mathcal{A}s_0'$ and DA_0'.
4. Obtain the MGU τ of the literals on which to resolve.
5. Show that ordered resolution on $C\mathcal{A}s_0'$ and DA_0' with τ as MGU is applicable.
6. Show that the resulting resolvent E_0 generalizes E with substitution θ.

In step 1, suitable clauses must be chosen so that $S\ (C\mathcal{A}s_0\,!\,i)$ generalizes $S_M\ (C\mathcal{A}s\,!\,i)$, for $0 \leq i < n$, and $S\ DA_0$ generalizes $S_M\ DA$. By the definition of S_M, this is always possible. In step 2, we choose the literals to resolve upon in the first-order inference depending on the selection on the ground inference. If some literals are selected in DA,

we define As_0 as the selected literals in DA_0, such that $(As_0 \,!\, i) \cdot \eta = As \,!\, i$ for each i. Otherwise, As must be a singleton list containing some atom A, and we define As_0 as the singleton list consisting of an arbitrary $A_0 \in DA_0$ such that $A_0 \cdot \eta = A$. Step 3 may seem straightforward until one realizes that renaming variables can in principle influence selection. To rule this out, our lemma assumes stability under renaming: $S\ (C \cdot \rho) = S\ C \cdot \rho$ for any renaming substitution ρ and clause C. This requirement seems natural, but it is not mentioned in the chapter.

The above choices allow us to perform steps 4 to 6. In the chapter, the authors assume that the obtained $C\mathcal{A}s_0$ and DA_0 are standardized apart from each other as well as their conclusion E_0. This means that they can obtain a single ground substitution μ that connect $C\mathcal{A}s_0$, DA_0, E_0 to $C\mathcal{A}s$, DA, E. By contrast, we provide separate substitutions ηs, η, θ for the different side premises, the main premise, and the conclusion.

7 A First-Order Prover

Modern resolution provers interleave inference steps with steps that delete or reduce (simplify) clauses. In their Section 4.3, Bachmair and Ganzinger introduce the nondeterministic abstract prover RP that works on triples of clause sets and that generalizes the Otter-style and DISCOUNT-style loops. RP's core rule, called inference computation, performs first-order ordered resolution as described above; the other rules delete or reduce clauses or move them between clause sets. We formalize RP and prove it complete assuming a fair strategy (`FO_Ordered_Resolution_Prover.thy`).

Abstract First-Order Prover. The RP prover is a relation $\rightsquigarrow$ on states of the form $(\mathcal{N}, \mathcal{P}, \mathcal{O})$, where $\mathcal{N}$ is the set of *new clauses*, $\mathcal{P}$ is the set of *processed clauses*, and $\mathcal{O}$ is the set of *old clauses*. RP's formal definition is very close to the original formulation:

inductive $\rightsquigarrow$ :: *'a state* $\Rightarrow$ *'a state* $\Rightarrow$ *bool* **where**
  Neg $A \in C \Longrightarrow$ Pos $A \in C \Longrightarrow (\mathcal{N} \cup \{C\}, \mathcal{P}, \mathcal{O}) \rightsquigarrow (\mathcal{N}, \mathcal{P}, \mathcal{O})$
| $D \in \mathcal{P} \cup \mathcal{O} \Longrightarrow$ subsumes $D\ C \Longrightarrow (\mathcal{N} \cup \{C\}, \mathcal{P}, \mathcal{O}) \rightsquigarrow (\mathcal{N}, \mathcal{P}, \mathcal{O})$
| $D \in \mathcal{N} \Longrightarrow$ strictly_subsumes $D\ C \Longrightarrow (\mathcal{N}, \mathcal{P} \cup \{C\}, \mathcal{O}) \rightsquigarrow (\mathcal{N}, \mathcal{P}, \mathcal{O})$
| $D \in \mathcal{N} \Longrightarrow$ strictly_subsumes $D\ C \Longrightarrow (\mathcal{N}, \mathcal{P}, \mathcal{O} \cup \{C\}) \rightsquigarrow (\mathcal{N}, \mathcal{P}, \mathcal{O})$
| $D \in \mathcal{P} \cup \mathcal{O} \Longrightarrow$ reduces $D\ C\ L \Longrightarrow (\mathcal{N} \cup \{C \uplus \{L\}\}, \mathcal{P}, \mathcal{O}) \rightsquigarrow (\mathcal{N} \cup \{C\}, \mathcal{P}, \mathcal{O})$
| $D \in \mathcal{N} \Longrightarrow$ reduces $D\ C\ L \Longrightarrow (\mathcal{N}, \mathcal{P} \cup \{C \uplus \{L\}\}, \mathcal{O}) \rightsquigarrow (\mathcal{N}, \mathcal{P} \cup \{C\}, \mathcal{O})$
| $D \in \mathcal{N} \Longrightarrow$ reduces $D\ C\ L \Longrightarrow (\mathcal{N}, \mathcal{P}, \mathcal{O} \cup \{C \uplus \{L\}\}) \rightsquigarrow (\mathcal{N}, \mathcal{P} \cup \{C\}, \mathcal{O})$
| $(\mathcal{N} \cup \{C\}, \mathcal{P}, \mathcal{O}) \rightsquigarrow (\mathcal{N}, \mathcal{P} \cup \{C\}, \mathcal{O})$
| $(\{\}, \mathcal{P} \cup \{C\}, \mathcal{O}) \rightsquigarrow$ (concl_of ‘ infers_between $\mathcal{O}\ C, \mathcal{P}, \mathcal{O} \cup \{C\}$)

The rules correspond, respectively, to tautology deletion, forward subsumption, backward subsumption in $\mathcal{P}$ and $\mathcal{O}$, forward reduction, backward reduction in $\mathcal{P}$ and $\mathcal{O}$, clause processing, and inference computation.

Initially, $\mathcal{N}$ consists of the problem clauses and the other two sets are empty. Clauses in $\mathcal{N}$ are reduced using $\mathcal{P} \cup \mathcal{O}$, or even deleted if they are tautological or subsumed by $\mathcal{P} \cup \mathcal{O}$; conversely, $\mathcal{N}$ can be used for reducing or subsuming clauses in $\mathcal{P} \cup \mathcal{O}$. Clauses eventually move from $\mathcal{N}$ to $\mathcal{P}$, one at a time. As soon as $\mathcal{N}$ is empty, a clause from $\mathcal{P}$ is selected to move to $\mathcal{O}$. Then all possible resolution inferences between this given clause and the clauses in $\mathcal{O}$ are computed and put in $\mathcal{N}$, closing the loop.

The subsumption and reduction rules depend on the following predicates:

$$\begin{aligned}
\mathsf{subsumes}\ D\ C &\Leftrightarrow \exists\sigma.\ D\cdot\sigma \subseteq C\\
\mathsf{strictly_subsumes}\ D\ C &\Leftrightarrow \mathsf{subsumes}\ D\ C \wedge \neg\ \mathsf{subsumes}\ C\ D\\
\mathsf{reduces}\ D\ C\ L &\Leftrightarrow \exists D'\ L'\ \sigma.\ D = D' \uplus \{L'\} \wedge -L = L'\cdot\sigma \wedge D'\cdot\sigma \subseteq C
\end{aligned}$$

The definition of the set $\mathsf{infers\_between}\ \mathcal{O}\ C$, on which inference computation depends, is more subtle. In the chapter, the set of inferences between C and $\mathcal{O}$ consists of all inferences from $\mathcal{O} \cup \{C\}$ that have C as *exactly one* of their premises. This, however, leads to an incomplete prover, because it ignores inferences that need multiple copies of C. For example, assuming a maximal selection function, the resolution inference

$$\frac{\mathsf{p} \quad \mathsf{p} \quad \neg\mathsf{p} \vee \neg\mathsf{p}}{\bot}$$

is possible. Yet if the clause $\neg\mathsf{p} \vee \neg\mathsf{p}$ reaches $\mathcal{O}$ earlier than p, the inference would not be performed. This counterexample requires ternary resolution, but there also exists a more complicated one for binary resolution, where both premises are the same clause. Consider the clause set containing

$$(1)\ \mathsf{q}(\mathsf{a},\mathsf{c},\mathsf{b}) \qquad (2)\ \neg\mathsf{q}(x,y,z) \vee \mathsf{q}(y,z,x) \qquad (3)\ \neg\mathsf{q}(\mathsf{b},\mathsf{a},\mathsf{c})$$

and an order $>$ on atoms such that $\mathsf{q}(\mathsf{c},\mathsf{b},\mathsf{a}) > \mathsf{q}(\mathsf{b},\mathsf{a},\mathsf{c}) > \mathsf{q}(\mathsf{a},\mathsf{c},\mathsf{b})$. Inferences between (1) and (2) or between (2) and (3) are impossible due to order restrictions. The only possible inference involves two copies of (2):

$$\frac{\neg\mathsf{q}(x,y,z) \vee \mathsf{q}(y,z,x) \quad \neg\mathsf{q}(x',y',z') \vee \mathsf{q}(y',z',x')}{\neg\mathsf{q}(x,y,z) \vee \mathsf{q}(z,x,y)}$$

From the conclusion, we derive $\neg\mathsf{q}(\mathsf{a},\mathsf{c},\mathsf{b})$ by (3) and $\bot$ by (1). This incompleteness is a severe flaw, although it is probably just an oversight.

Projection to Theorem Proving Process. On the first-order level, a derivation can be expressed as a lazy list $\mathcal{S}s$ of states, or as three parallel lazy lists $\mathcal{N}s$, $\mathcal{P}s$, $\mathcal{O}s$. The limit state of a derivation $\mathcal{S}s$ is defined as $\mathsf{Liminf}\ \mathcal{S}s = (\mathsf{Liminf}\ \mathcal{N}s, \mathsf{Liminf}\ \mathcal{P}s, \mathsf{Liminf}\ \mathcal{O}s)$, where Liminf on the right-hand side is as in Section 5.

Bachmair and Ganzinger use the completeness of ground resolution to prove RP complete. The first step is to show that first-order derivations can be projected down to theorem proving processes on the ground level. This corresponds to Lemma 4.10. Adapted to our conventions, its statement is as follows:

> If $\mathcal{S} \rightsquigarrow \mathcal{S}'$, then $\mathsf{grounding\_of}\ \mathcal{S} \rhd^* \mathsf{grounding\_of}\ \mathcal{S}'$, with $\rhd$ based on some extension of ordered resolution with selection function S and the standard redundancy criterion $(\mathcal{R}_\mathcal{F}, \mathcal{R}_\mathcal{I})$.

This raises some questions: (1) Exactly which instance of the calculus are we extending? (2) Which calculus extension should we use? (3) How can we repair the mismatch between $\rhd^*$ in the lemma statement and $\rhd$ where the lemma is invoked?

Regarding question (1), it is not clear which selection function to use. Is the function the same S as in the definition of RP or is it arbitrary? It takes a close inspection of the proof of Lemma 4.13, where Lemma 4.10 is invoked, to find out that the selection function used there is $S_{\mathsf{Liminf}\ \mathcal{O}s}$.

Regarding question (2), the phrase "some extension" is cryptic. It suggests an existential reading, and from the context it would appear that a standard extension (Section 5) is meant. However, neither the lemma's proof nor the context where it is invoked supplies the desired existential witness. A further subtlety is that the witness should be independent of $\mathcal{S}$ and $\mathcal{S}'$, so that transitions can be joined to form a single theorem proving derivation. Our approach is to let $\rhd$ be the extension consisting of all *sound* derivations: $\Gamma = \{(C, D, E) \mid \forall I.\ I \models C \cup \{D\} \Longrightarrow I \models E\}$. This also eliminates the need for Bachmair and Ganzinger's subsumption resolution rule, a special calculus rule that is, from what we understand, implicitly used in the proof of Lemma 4.10 for the subcases associated with RP's reduction rules.

As for question (3), the need for $\rhd^*$ instead of $\rhd$ arises because one of the cases requires a combination of deduction and deletion, which Bachmair and Ganzinger model as separate transitions. By merging the two transitions (Section 5), we avoid the issue altogether and can use $\rhd$ in the formal counterpart of Lemma 4.10.

With these issues resolved, we can prove Lemma 4.10 for single steps and extend it to entire derivations:

lemma *RP_ground_derive*: $\mathcal{S} \rightsquigarrow \mathcal{S}' \Longrightarrow \mathsf{grounding\_of}\ \mathcal{S} \rhd \mathsf{grounding\_of}\ \mathcal{S}'$

lemma *RP_ground_derive_chain*:
$\mathsf{chain}\ (\rightsquigarrow)\ \mathcal{S}s \Longrightarrow \mathsf{chain}\ (\rhd)\ (\mathsf{lmap}\ \mathsf{grounding\_of}\ \mathcal{S}s)$

The lmap function applies its first argument elementwise to its second argument.

Fairness and Clause Movement. From a given initial state $(\mathcal{N}_0, \{\}, \{\})$, many derivations are possible, reflecting RP's nondeterminism. In some derivations, we could leave a crucial clause in $\mathcal{N}$ or $\mathcal{P}$ without ever reducing it or moving it to $\mathcal{O}$, and then fail to derive $\bot$ even if $\mathcal{N}_0$ is unsatisfiable. For this reason, refutational completeness is guaranteed only for fair derivations. These are defined as derivations such that $\mathsf{Liminf}\ \mathcal{N}s = \mathsf{Liminf}\ \mathcal{P}s = \{\}$, guaranteeing that no clause will stay forever in $\mathcal{N}$ or $\mathcal{P}$.

Fairness is expressed by the fair_state_seq predicate, which is distinct from the fair_clss_seq predicate presented in Section 5. In particular, Theorem 4.3 is used in neither the informal nor the formal proof, and appears to play a purely pedagogic role in the chapter. For the rest of this section, we fix a lazy list of states $\mathcal{S}s$, and its projections $\mathcal{N}s$, $\mathcal{P}s$, and $\mathcal{O}s$, such that $\mathsf{chain}\ (\rightsquigarrow)\ \mathcal{S}s$, $\mathsf{fair\_state\_seq}\ \mathcal{S}s$, and $\mathsf{lhd}\ \mathcal{O}s = \{\}$.

Thanks to fairness, any nonredundant clause C in $\mathcal{S}s$'s projection to the ground level eventually ends up in $\mathcal{O}$ and stays there. This is proved informally as Lemma 4.11, but again there are some difficulties. The vagueness concerning the selection function can be resolved as for Lemma 4.10, but there is another, deeper flaw.

Bachmair and Ganzinger's proof idea is as follows. By hypothesis, the ground clause C must be an instance of a first-order clause D in $\mathcal{N}s\,!\,j \cup \mathcal{P}s\,!\,j \cup \mathcal{O}s\,!\,j$ for some index j. If $C \in \mathcal{N}s\,!\,j$, then by nonredundancy of C, fairness of the derivation, and Lemma 4.10, there must exist a clause D' that generalizes C in $\mathcal{P}s\,!\,l \cup \mathcal{O}s\,!\,l$ for some $l > j$. By a similar argument, if D' belongs to $\mathcal{P}s\,!\,l$, it will be in $\mathcal{O}s\,!\,l'$ for some $l' > l$,

and finally in all $\mathcal{O}s\,!\,k$ with $k \geq l'$. The flaw is that backward subsumption can delete D' without moving it to $\mathcal{O}$. The subsumer clause would then be a strictly more general version of D' (and of the ground clause C).

Our solution is to choose D, and consequently D', such that it is minimal, with respect to subsumption, among the clauses that generalize C in the derivation. This works because strict subsumption is well founded—which we also proved, by reduction to a well-foundedness result about the strict generalization relation on first-order terms, included in IsaFoR [13, Section 2]. By minimality, D' cannot be deleted by backward subsumption. This line of reasoning allows us to prove Lemma 4.11, where $\mathcal{O}$_of extracts the $\mathcal{O}$ component of a state:

lemma *fair_imp_Liminf_minus_Rf_subset_ground_Liminf_state*:
$Gs = \mathsf{lmap}\ \mathsf{grounding\_of}\ \mathcal{S}s \Longrightarrow$
$\mathsf{Liminf}\ Gs - \mathcal{R}_{\mathcal{F}}\ (\mathsf{Liminf}\ Gs) \subseteq \mathsf{grounding\_of}\ (\mathcal{O}\mathsf{\_of}\ (\mathsf{Liminf}\ \mathcal{S}s))$

Completeness. Once we have brought Lemmas 4.10, 4.11, and 4.12 into a suitable shape, the main completeness result, Theorem 4.13, is not difficult to formalize:

theorem *RP_saturated_if_fair*: $\mathsf{saturated\_upto}\,(\mathsf{Liminf}\,(\mathsf{lmap}\,\mathsf{grounding\_of}\,\mathcal{S}s))$

corollary *RP_complete_if_fair*:
$\neg\ \mathsf{sat}\ (\mathsf{grounding\_of}\ (\mathsf{lhd}\ \mathcal{S}s)) \Longrightarrow \{\} \in \mathcal{O}\mathsf{\_of}\ (\mathsf{Liminf}\ \mathcal{S}s)$

A crucial point that is not clear from the text is that we must always use the selection function S on the first-order level and $S_{\mathsf{Liminf}\ \mathcal{O}s}$ on the ground level. Another noteworthy part of the proof is the passage "Liminf Gs (and hence Liminf $\mathcal{S}s$) contains the empty clause" (using our notations). Obviously, if grounding_of (Liminf $\mathcal{S}s$) contains $\bot$, then Liminf $\mathcal{S}s$ must as well. However, the authors do not explain the step from Liminf Gs, the limit of the grounding, to grounding_of (Liminf $\mathcal{S}s$), the grounding of the limit. Fortunately, by Lemma 4.11, the latter contains all the nonredundant clauses of the former, and the empty clause is nonredundant. Hence the informal argument is fundamentally correct.

8 Discussion and Related Work

Bachmair and Ganzinger cover a lot of ground in a few pages. We found much of the material straightforward to formalize: it took us about two weeks to reach their Section 4.3, which introduces the RP prover. By contrast, we needed months to fully understand and formalize that section. While the *Handbook* chapter succeeds at conveying the key ideas at the propositional level, the lack of rigor makes it difficult to develop a deep understanding of ordered resolution proving on first-order clauses.

There are several reasons why Section 4.3 did not lend itself easily to a formalization. The proofs often depend on lemmas and theorems from previous sections without explicitly mentioning them. The lemmas and proofs do not quite fit together. And while the general idea of the proofs stands up, they have many confusing flaws that must be repaired. Our methodology involved the following steps: (1) rewrite the informal proofs to a handwritten pseudo-Isabelle; (2) fill in the gaps, emphasizing which lemmas are used where; (3) turn the pseudo-Isabelle to real Isabelle, but with **sorry** placeholders

for the proofs; and (4) replace the **sorry**s with proofs. Progress was not always linear. As we worked on each step, more than once we discovered an earlier mistake.

The formalization helps us answer questions such as, "Is effectiveness of ordered resolution (Lemma 3.13) actually needed, and if so, where?" It also allows us to track definitions and hypotheses precisely, so that we always know the scope and meaning of every definition, lemma, or theorem. If a hypothesis appears too strong or superfluous, we can try to rephrase or eliminate it; the proof assistant tells us where the proof breaks.

Starting from RP, we could refine it to obtain an efficient imperative implementation, following the lines of Fleury, Blanchette, and Lammich's verified SAT solver with the two-watched-literals optimization [12]. However, this would probably involve a huge amount of work. To increase provers' trustworthiness, a more practical approach is to have them generate detailed proofs. Such output can be independently reconstructed using a proof assistant's inference kernel. This is the approach implemented in Sledgehammer [8], which integrates automatic provers in Isabelle. Formalized metatheory could in principle be used to deduce a formula's satisfiability from a finite saturation.

We found Isabelle/HOL eminently suitable to this kind of formalization work. Its logic—based on classical simple type theory—balances expressiveness and automatability. We benefited from many features of the system, including codatatypes [5], Isabelle/jEdit [28], the Isar proof language [27], locales [4], and Sledgehammer [8]. It is perhaps indicative of the maturity of theorem proving technology that most of the issues we encountered were unrelated to Isabelle. The main challenge was to understand the informal proof well enough to design suitable locale hierarchies and state the definitions and lemmas precisely, and correctly.

Formalizing the metatheory of logic and deduction is an enticing proposition for many researchers. Two recent, independent developments are particularly pertinent. Peltier [17] proved static completeness of a variant of the superposition calculus in Isabelle/HOL. Since superposition generalizes ordered resolution, his result subsumes our static completeness theorem. It would be interesting to extend his formal development to obtain a verified superposition prover. We could also consider calculus extensions such as polymorphism [11, 25], type classes [25], and AVATAR [24]. Hirokawa et al. [13] formalized, also in Isabelle/HOL, an abstract Knuth–Bendix completion procedure as well as ordered (unfailing) completion [1]. Superposition combines ordered resolution (to reason about clauses) and ordered completion (to reason about equality).

The literature contains many other formalized completeness proofs. Early work was carried out by Shankar [22] and Persson [18]. Some of our own efforts are also related: completeness of unordered resolution using semantic trees by Schlichtkrull [20]; completeness of a Gentzen system by Blanchette, Popescu, and Traytel [9]; and completeness of CDCL by Blanchette, Fleury, and Weidenbach [6]. We refer to our earlier papers for further discussions of related work.

9 Conclusion

We presented a formal proof that captures the core of Bachmair and Ganzinger's *Handbook* chapter on resolution theorem proving. For all its idiosyncrasies, the chapter withstood the test of formalization, once we had added self-inferences to the RP prover. Given that the text is a basic building block of automated reasoning, we believe there is

value in clarifying its mathematical content for the next generations of researchers. We hope that our work will be useful to the editors of a future revision of the *Handbook*.

Formalization of the metatheory of logical calculi is one of the many connections between automatic and interactive theorem proving. We expect to see wider adoption of proof assistants by researchers in automated reasoning, as a convenient way to develop metatheory. By building formal libraries of standard results, we aim to make it easier to formalize state-of-the-art research as it emerges. We also see potential uses of formal proofs in teaching automated reasoning, inspired by the use of proof assistants in courses on the semantics of programming languages [15, 19].

Acknowledgment. Christoph Weidenbach discussed Bachmair and Ganzinger's chapter with us on many occasions and hosted Schlichtkrull at the Max-Planck-Institut in Saarbrücken. Christian Sternagel and René Thiemann answered our questions about IsaFoR. Mathias Fleury, Florian Haftmann, and Tobias Nipkow helped enrich and reorganize Isabelle's multiset library. Mathias Fleury, Robert Lewis, Mark Summerfield, Sophie Tourret, and the anonymous reviewers suggested many textual improvements.

Blanchette was partly supported by the Deutsche Forschungsgemeinschaft (DFG) project Hardening the Hammer (grant NI 491/14-1). He also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka). Traytel was partly supported by the DFG program Program and Model Analysis (PUMA, doctorate program 1480).

References

[1] Bachmair, L., Dershowitz, N., Plaisted, D.A.: Completion without failure. In: Aït-Kaci, H., Nivat, M. (eds.) Rewriting Techniques—Resolution of Equations in Algebraic Structures, vol. 2, pp. 1–30. Academic Press (1989)

[2] Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. 4(3), 217–247 (1994)

[3] Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 19–99. Elsevier and MIT Press (2001)

[4] Ballarin, C.: Locales: A module system for mathematical theories. J. Autom. Reason. 52(2), 123–153 (2014)

[5] Biendarra, J., Blanchette, J.C., Bouzy, A., Desharnais, M., Fleury, M., Hölzl, J., Kuncar, O., Lochbihler, A., Meier, F., Panny, L., Popescu, A., Sternagel, C., Thiemann, R., Traytel, D.: Foundational (co)datatypes and (co)recursion for higher-order logic. In: Dixon, C., Finger, M. (eds.) FroCoS 2017. LNCS, vol. 10483, pp. 3–21. Springer (2017)

[6] Blanchette, J.C., Fleury, M., Lammich, P., Weidenbach, C.: A verified SAT solver framework with learn, forget, restart, and incrementality. Accepted in J. Autom. Reason.

[7] Blanchette, J.C., Fleury, M., Traytel, D.: Nested multisets, hereditary multisets, and syntactic ordinals in Isabelle/HOL. In: Miller, D. (ed.) FSCD 2017. LIPIcs, vol. 84, pp. 11:1–11:18. Schloss Dagstuhl—Leibniz-Zentrum für Informatik (2017)

[8] Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.: Hammering towards QED. J. Formaliz. Reason. 9(1), 101–148 (2016)

[9] Blanchette, J.C., Popescu, A., Traytel, D.: Soundness and completeness proofs by coinductive methods. J. Autom. Reason. 58(1), 149–179 (2017)

[10] Brand, D.: Proving theorems with the modification method. SIAM J. Comput. 4(4), 412–430 (1975)

[11] Cruanes, S.: Logtk: A logic toolkit for automated reasoning and its implementation. In: Schulz, S., de Moura, L., Konev, B. (eds.) PAAR-2014. EPiC Series in Computing, vol. 31, pp. 39–49. EasyChair (2014)
[12] Fleury, M., Blanchette, J.C., Lammich, P.: A verified SAT solver with watched literals using Imperative HOL. In: Andronick, J., Felty, A.P. (eds.) CPP 2018. pp. 158–171. ACM (2018)
[13] Hirokawa, N., Middeldorp, A., Sternagel, C., Winkler, S.: Infinite runs in abstract completion. In: Miller, D. (ed.) FSCD 2017. LIPIcs, vol. 84, pp. 19:1–19:16. Schloss Dagstuhl—Leibniz-Zentrum für Informatik (2017)
[14] Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 371–443. Elsevier and MIT Press (2001)
[15] Nipkow, T.: Teaching semantics with a proof assistant: No more LSD trip proofs. In: Kuncak, V., Rybalchenko, A. (eds.) VMCAI 2012. LNCS, vol. 7148, pp. 24–38. Springer (2012)
[16] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
[17] Peltier, N.: A variant of the superposition calculus. Archive of Formal Proofs 2016 (2016), https://www.isa-afp.org/entries/SuperCalc.shtml
[18] Persson, H.: Constructive completeness of intuitionistic predicate logic—a formalisation in type theory (1996)
[19] Pierce, B.C.: Lambda, the ultimate TA: Using a proof assistant to teach programming language foundations. In: Hutton, G., Tolmach, A.P. (eds.) ICFP 2009. pp. 121–122. ACM (2009)
[20] Schlichtkrull, A.: Formalization of the resolution calculus for first-order logic. Accepted in J. Autom. Reason.
[21] Schlichtkrull, A., Blanchette, J.C., Traytel, D., Waldmann, U.: Formalizing Bachmair and Ganzinger's ordered resolution prover (technical report). Technical report (2018), http://matryoshka.gforge.inria.fr/pubs/rp_report.pdf
[22] Shankar, N.: Towards mechanical metamathematics. J. Autom. Reason. 1(4), 407–434 (1985)
[23] Thiemann, R., Sternagel, C.: Certification of termination proofs using CeTA. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 452–468. Springer (2009)
[24] Voronkov, A.: AVATAR: The architecture for first-order theorem provers. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 696–710. Springer (2014)
[25] Wand, D.: Polymorphic+typeclass superposition. In: Schulz, S., de Moura, L., Konev, B. (eds.) PAAR-2014. EPiC Series in Computing, vol. 31, pp. 105–119. EasyChair (2014)
[26] Weidenbach, C.: Combining superposition, sorts and splitting. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. II, pp. 1965–2013. Elsevier and MIT Press (2001)
[27] Wenzel, M.: Isabelle/Isar—a generic framework for human-readable proof documents. In: Matuszewski, R., Zalewska, A. (eds.) From Insight to Proof: Festschrift in Honour of Andrzej Trybulec, Studies in Logic, Grammar, and Rhetoric, vol. 10(23). University of Białystok (2007)
[28] Wenzel, M.: Isabelle/jEdit—a prover IDE within the PIDE framework. In: Jeuring, J., Campbell, J.A., Carette, J., Reis, G.D., Sojka, P., Wenzel, M., Sorge, V. (eds.) CICM 2012. LNCS, vol. 7362, pp. 468–471. Springer (2012)
[29] Zhang, H., Kapur, D.: First-order theorem proving using conditional rewrite rules. In: Lusk, E.L., Overbeek, R.A. (eds.) CADE-9. LNCS, vol. 310, pp. 1–20. Springer (1988)